# Pawan Chaudhary

## 1 Understanding Data Strucutes

A mutable object can be altered after it is created, where the changes occur in the same memory address whereas an immutable object cannot be altered after it is created, which means in order to make changes the immutable object will have to create a new object.

Lists are a mutable data structure which allows users to make changes after its creation. For instance, a to-do list allows us to add, delete or modify its contents.

Tuples are an immutable data structure which remains fixed after its creation allowing no alteration. For instance, tuples are generally used to store read-only data like mathematical constants.

## 2 Working with Data Structures

I learned to find the mean nad median easily by importing NumPy. Here is the reference site: https://www.w3schools.com/python/python_ml_mean_median_mode.asp

In [ ]:
```
chaudhary_sales_q1 = [1200, 850, 1150, 950]
print(f"{chaudhary_sales_q1}")
```
```
[1200, 850, 1150, 950]
```

In [ ]:
```
import numpy as np
```

In [ ]:
```
# calculating mean

mean_sales_q1 = np.mean(chaudhary_sales_q1)
print(f"The mean value of sales q1 is: {mean_sales_q1}")
```
```
The mean value of Sales q1 is: 1037.5
```

In [12]:
```
# calculating median

median_sales_q1 = np.median(chaudhary_sales_q1)
print(f"The median value of sales q1 is: {median_sales_q1}")
```
```
The median value of sales q1 is: 1050.0
```

In [15]:
```
# adjusting values

adjusted_sales_q1 = chaudhary_sales_q1.copy()
adjusted_sales_q1.append(1300)
print(f"{adjusted_sales_q1}")
```

```
[1200, 850, 1150, 950, 1300, 1300]
```

In [23]:
```
# re-calculated mean

new_mean_sales_q1 = np.mean(adjusted_sales_q1)
print(f"The new mean value of sales q1 is: {new_mean_sales_q1}")
```

```
The new mean value of sales q1 is: 1125.0
```

In [24]:
```
# re-calculated median

new_median_sales_q1 = np.median(adjusted_sales_q1)
print(f"The new median value of sales q1 is: {new_median_sales_q1}")
```

```
The new median value of sales q1 is: 1175.0
```

## 3 Conditional Logic

In [ ]:
```
performance_five_employee = [78, 85, 62, 90, 88]
print(f"{performance_five_employee}")
```

```
[78, 85, 62, 90, 88]
```

In [36]:
```
# function for returning distribution of performance ratings

def distribution(performance_five_employee):
    performance_distribution = {
        "Excellent": 0,
        "Good": 0,
        "Satisfactory": 0,
        "Needs Improvement": 0,
        "Unsatisfactory": 0,
    }


    for i in range (len(performance_five_employee)):
        if performance_five_employee[i] >= 90:
            performance_distribution["Excellent"] += 1
        elif 80 <= performance_five_employee[i] <= 89:
            performance_distribution["Good"] += 1
        elif 70 <= performance_five_employee[i] <= 79:
            performance_distribution["Satisfactory"] += 1
        elif 60 <= performance_five_employee[i] <= 69:
            performance_distribution["Needs Improvement"] += 1
        else:
            performance_distribution["Unsatisfactory"] += 1

    return performance_distribution
```

In [35]:
```
ratings = distribution(performance_five_employee)
print(f"{ratings}")
```

```
{'Excellent': 1, 'Good': 2, 'Satisfactory': 1, 'Needs Improvement': 1, 'Unsatisfacto
ry': 0}
```

## 4 Data Cleaning

I learned how to remove all the punctuations from geeksforgeeks. Here is the reference site: https://www.geeksforgeeks.org/python/python-remove-punctuation-from-string/

In [ ]:
```
#1 Removing punctuation

import re

feedback = "Excellent! service; but the product is too expensive. Great, customer c
cleaned_feedback = re.sub(r'[^\w\s]', '', feedback)
print(cleaned_feedback)
```

```
Excellent service but the product is too expensive Great customer care
```

In [49]:
```
#2 Convering all text to lowercase

cleaned_lower_feedback = cleaned_feedback.lower()
print(f"{cleaned_lower_feedback}")
```

```
excellent service but the product is too expensive great customer care
```

In [52]:
```
#3 Splitting the cleaned text into individual words

cleaned_lower_split_feedback = cleaned_lower_feedback.split()
print(f"{cleaned_lower_split_feedback}")
```

```
['excellent', 'service', 'but', 'the', 'product', 'is', 'too', 'expensive', 'great',
'customer', 'care']
```

In [54]:
```
# Identifying unique words
# using set() as it helps it helps removing duplicate values

cleaned_final = set(cleaned_lower_split_feedback)
print(f"Final cleaned feedback with unique words is: {cleaned_final}")
```

```
Final cleaned feedback with unique words is: {'care', 'great', 'too', 'expensive',
'the', 'product', 'service', 'but', 'customer', 'is', 'excellent'}
```

## 5 Extra

In [57]:
```
# department dictionary

departments = {
    "Marketing": [11000, 15000, 18000],
    "Development": [28000, 11500, 17090],
    "Operations": [56000, 52000, 49000],
    "Human Resources": [9000, 4000, 7000],
    "Information Technology": [70000, 68000, 67000],
}

print(f"Department and Expenses Dictionary: {departments}")
```

```
Department and Expenses Dictionary: {'Marketing': [11000, 15000, 18000], 'Developmen
t': [28000, 11500, 17090], 'Operations': [56000, 52000, 49000], 'Human Resources':
[9000, 4000, 7000], 'Information Technology': [70000, 68000, 67000]}
```

In [66]:
```
# function to calculate sum and average


def expenses(departments):
    for deparment, expenses in departments.items():
        total_sales = sum(expenses)
        average_sales = total_sales/len(expenses)
        print(f"Department: {deparment}")
        print(f"Total Sales: {total_sales}")
        print(f"Average Sales: {average_sales}")
        print(f"--------------------------------")


expenses(departments)
```

```
Department: Marketing
Total Sales: 44000
Average Sales: 14666.666666666666
--------------------------------
Department: Development
Total Sales: 56590
Average Sales: 18863.333333333332
--------------------------------
Department: Operations
Total Sales: 157000
Average Sales: 52333.333333333336
--------------------------------
Department: Human Resources
Total Sales: 20000
Average Sales: 6666.666666666667
--------------------------------
Department: Information Technology
Total Sales: 205000
Average Sales: 68333.33333333333
--------------------------------
```

In [ ]:
```
# accessing marketing expenses

marketing_expense = departments["Marketing"]
print(f"{marketing_expense}")
```

```
[11000, 15000, 18000]
```

In [ ]:
```
# updating marketing expenses

new_marketing_expense=[]
for exp in marketing_expense:
    increased_expense = exp * 1.15
    new_marketing_expense.append(increased_expense)

print(f"Updated Marketing Expenses: {new_marketing_expense}")
```

```
Updated Marketing Expenses: [12649.999999999998, 17250.0, 20700.0]
```